REMARKS

Claims 1-19 were pending as of the action mailed on October 10, 2006. Claims 1, 13, and 17 are in independent form.

Claims 1, 3, 5-7, 13-19 are being amended. Claims 20-27 are new. New claim 25 is in independent form. No new matter has been added.

To expedite prosecution, and without prejudice, the applicant has amended the preamble of claim 1 in anticipation of the Office's now standard section 101 rejection of claims directed to information carriers that can include propagated signals.

Reexamination and reconsideration of the action are requested in light of the foregoing amendments and the following remarks.

Claim Objections

The examiner's objections to the claims have been addressed by amending claims 14-16 and 18-19. The applicant respectfully submits that the objections have been overcome.

Rejections under Section 102

Claims 1-19 were rejected under 35 U.S.C. § 102(e) as allegedly anticipated by U.S. 2003/0217354 ("Bates").

Claim 1. The examiner rejected claim 1 stating that

> Bates discloses a computer program product (pages 6-7, computer readable medium), tangibly embodied on an information carrier, comprising instructions operable to cause data processing apparatus to:
> establish any number of checkpoints in a first computer program (see for example page 3, [0040], lines 3-9); and
> include each checkpoint in a checkpoint group, wherein each checkpoint group can include any number of the checkpoints regardless of where the checkpoints are in the first computer program (see for example FIG. 2, item 150, "breakpoint table").

Claim 1 has been amended to recite that the claimed program product is operable to "assign each checkpoint in the plurality of checkpoints to a checkpoint group without regard to the program structure of the first computer program". Thus, as claim 1 further recites, "the

structure of checkpoint groups is independent of the program structure of the first computer program."

Neither is true of Bates. In Bates, the grouping of checkpoints is based on the structure of the program, because it is based on "scope". For example, Bates teaches:

> An operation herein is generally any operation affecting the breakpoints within a selected scope. In one embodiment, the user selects a line for the purpose of selecting a scope that contains the line. In response, the debugger determines an innermost scope that contains the line. The innermost scope therefore becomes the selected scope. The debugger then determines all the breakpoints set within the selected scope. The user then selects an operation to be executed on all the breakpoints within the selected scope. [0029]

Scope is a defined term in Bates: "As used herein, the term 'scope' refers to a software language construct of the source code for the computer program 120. For example, a while loop defines a scope." [0045] Thus, for example, when "the Group operation option is selected, then all the breakpoints within the selected scope will be collected into one breakpoint group." [0052] Similarly, when the Enable operation option is selected, "the breakpoints within the selected scope are enabled (step 1220)." [0062]

In addition, although the examiner points to Bates as teaching that "each checkpoint group can include any number of the checkpoints regardless of where the checkpoints are in the first computer program (see for example FIG. 2, item 150, 'breakpoint table'), " neither FIG. 2 nor the description of item 150, indicates that "each checkpoint group can include . . . checkpoints regardless of where the checkpoints are in the first computer program." In fact, item 150 and "breakpoint table" are mentioned only in [0039], which provides no description of the functionality of the breakpoint table to support the examiner's assertion.

For the foregoing reasons, the rejection of claim 1 and its dependent claims should be withdrawn.

Claim 5. The examiner rejected claim 5 stating that

> Bates further discloses the product of claim 4, wherein the control input further specifies a mode and the mode comprises one of:

activating checkpoints that are assertions to terminate on assertion failure (see for example FIG. 12B, step 1230, and related text);
activating checkpoints that are assertions to log status on assertion failure (see for example FIG. 12B, step 1230, and related text); and
activating checkpoints that are assertions to break in a debugger on assertion failure (see for example FIG. 12B, step 1230, and related text).

Claim 5 depends from claim 4 which depends from claim 1. For at least the reasons that apply to its base claims, the rejection of claim 5 should be withdrawn.

In addition, claim 5 has been amended to clarify that the claimed product has instructions to handle all three of the recited modes. Thus, claim 5 as amended recites that the instructions of the claimed program product include instructions to:

receive a control input that specifies a mode in which checkpoints that are assertions terminate on assertion failure;
receive a control input that specifies a mode in which checkpoints that are assertions log status on assertion failure; and
receive a control input that specifies a mode of activating checkpoints in which assertions break in a debugger on assertion failure

The cited portion of Bates does not teach or suggest these features. The only place in Bates where step 1230 is discussed appears to be paragraph [0063], which reads in its entirety as follows:

[0063] At step 1230, a determination is made as to whether the user has selected the Condition operation option. If so, a condition dialog box, such as the condition dialog box 1000 in FIG. 10, is displayed to prompt the user for a user-specified condition (step 1232). In one embodiment, the user is given the option to associate the user-specified condition with each breakpoint as either a conjunction with an existing condition of the breakpoint, a disjunction with the existing condition of the breakpoint, or a replacement of the existing condition. For instance, in FIG. 10, the user has entered "i>50" as the user-specified condition and has selected to associate the user-specified condition as a conjunction with the existing condition. Upon receiving the user-specified condition, the user-specified condition is associated with each breakpoint according to the manner in which the user-specified condition is to be associated (steps 1236-1280).

This text does not describe the modes recited in the claim.

For the foregoing additional reason, the rejection of claim 5 should be withdrawn.

Claim 6. The examiner rejected claim 6 stating that

> Bates further discloses the product of claim 4, further comprise instructions to:
> receive a control input specifying a scope (see for example FIG. 12A, steps1204, 1206, and related text).

Claim 6 depends from claim 4 which depends from claim 1. For at least the reasons that apply to its base claims, the rejection of claim 6 should be withdrawn.

In addition, claim 6 has been amended to recite that the product includes instructions to "receive a control input specifying that activating is to be performed only for a particular user of the first computer program," a limitation previously found in claim 7.

In rejecting claim 7, the examiner relied on "FIG. 12A, steps 1204, 1206, and related text." Steps 1204 and 1206 are cited only in paragraphs [0057] and [0058] of Bates, which read in their entirety as follows:

> [0057] Referring now to FIGS. 12A-C, a method 1200 of operating the debugger 123 in a manner consistent with embodiments of the present invention is shown. Upon receiving a debug event (step 1202), the debugger 123 determines whether the debug event is a selection of a line for the purpose of identifying or selecting the innermost scope that contains the selected line (step 1204). The selection may be made by various means, such as, right clicking on the selected line or left clicking on the selected line while holding down an Alt-key. If the debug event is not a selection of a line for the purpose of identifying the innermost scope, then the debug event is handled in an appropriate manner according to the design of the debugger 123 (step 1234). Examples for such a debug event include events for asking variable values, setting the breakpoints, and scrolling the screen up and down. Processing then returns to step 1202 at which the debugger 123 waits for the next debug event.
>
> [0058] On the other hand, if the debug event is a selection of a line for the purpose of identifying or selecting the innermost scope that contains the selected line, such as line 43 in FIG. 3, then the range of the innermost scope that contains the line is determined (step 1206). In one embodiment, if the innermost scope for the selected line cannot be determined, then processing will return to step 1202 at which the debugger 123 waits for the next debug event. In another embodiment, the range of the selected scope is

> determined from data, such as nesting data, generated by the compiler 120. In yet another embodiment, once the range of the selected scope is determined, the range is graphically represented by a scope range bar, such as the scope bar 310 shown in FIG. 3.

The cited passages do not teach the recited limitation that "activating is to be performed only for a particular user."

For the foregoing additional reason, the rejection of claim 6 should be withdrawn.

Claim 7. The examiner rejected claim 7 stating that

> Bates further discloses the product of claim 4, further comprise instructions to:
> receive a control input specifying a scope specifying that activating is to be performed only for a particular user of the first computer program, that activating is to be performed only for a particular server on which the first computer program is running, or that activating is to be performed globally (see for example FIG. 12A, steps 1204, 1206, and related text).

Claim 7 depends from claim 4 which depends from claim 1. For at least the reasons that apply to its base claims, the rejection of claim 7 should be withdrawn.

In addition, claim 7 has been amended to recite that the product includes instructions to

> receive a control input specifying that activating is to be performed only for a particular server on which the first computer program is running. (Emphasis added)

The portion of Bates on which the examiner relies in rejecting claim 7, namely "FIG. 12A, steps 1204, 1206, and related text," has just been reproduced in its entirety. As can readily be seen, the word "server" does not occur in this portion of Bates; nor does Bates describe activating a checkpoint group only for a particular server, as expressly recited in the claim.

For the foregoing additional reason, the rejection of claim 7 should be withdrawn.

Claim 13. The examiner rejected claim 13 stating that

> Bates discloses an apparatus (see for example FIG. 1, and related text), comprising:
> means for establishing any number of checkpoints in a computer program (see for example page 3, [0040], lines 3-9); and

means for including each checkpoint in a checkpoint group, wherein each checkpoint group can include any number of the checkpoints regardless of where the checkpoints are in the computer program (see for example FIG. 2, item 150, "breakpoint table").

This rejection corresponds to the rejection of claim 1.

For the reasons set forth above in reference to claim 1, the rejection of claim 13 should also be withdrawn.

Claim 17. The examiner rejected claim 17 stating that

Bates discloses a method comprising:
receiving a computer program having checkpoints each identified by a group identifier, each group identifier identifying checkpoints without limitation as to the location of the checkpoints in the computer program, each checkpoint being an assertion or a breakpoint (see for example page 3, [0040, lines 1-9); and
receiving user input to invoke checkpoints as a group according to their group identifiers (see for example page 3, [0043], "stop handler").

This rejection corresponds to the rejection of claim 1, with the additional consideration of the group identifiers.

The examiner relies on paragraph [0043] as teaching group identifiers in reference to stop handlers. This paragraph reads in its entirety as follows, with emphasis added:

[0043] After the commands are entered, the user provides an input that resumes execution of the program 120. During execution, control is returned to the debugger 123 via the stop handler 134. The stop handler 134 is a code segment that returns control to the appropriate user interface. In some implementations, execution of the program eventually results in an event causing a trap to fire (e.g., a breakpoint or watchpoint is encountered). Inserting and managing special op codes that cause these traps to fire is the responsibility of the breakpoint manager 130. When a trap fires, control is then returned to the debugger by the stop handler 134 and program execution is halted. The stop handler 134 then invokes the debug user interface 124 and may pass the results to the user interface 124. Alternatively, the results may be passed to the results buffer 136 to cache data for the user interface 124. In other embodiments, the user may input a command while the program is stopped, causing the debugger to run a desired debugging routine. Result values are then provided to the user via the user interface 124.

As is clear from the emphasized text, a stop handler is a code segment. Code segments are not inherently identifiers. Nothing in the cited paragraph teaches or suggests that the code segment is or can be used as an identifier for a group of checkpoints, as recited in the claim.

For the foregoing reason and the reasons set forth above in reference to claim 1, the rejection of claim 17 should be withdrawn.

Claim 25. Claim 25 is a new claim that includes limitation corresponding to limitations of claims 1 and 17 that have been discussed in the foregoing remarks.

For the reasons set forth above in reference to those limitations, claim 25 is allowable.

Remaining claims. The remaining claims depend from claims that are allowable for reasons set forth above.

Conclusion

For the foregoing reasons, the applicant submits that all the claims are in condition for allowance.

By responding in the foregoing remarks only to particular positions taken by the examiner, the applicant does not acquiesce with other positions that have not been explicitly addressed. In addition, the applicant's arguments for the patentability of a claim should not be understood as implying that no other reasons for the patentability of that claim exist.

The fee in the amount of $550.00 for additional claims are being paid concurrently herewith on the Electronic Filing System (EFS) by way of Deposit Account authorization. Please apply any charges or credits to deposit account 06-1050.

Respectfully submitted,

Date: 14 Dec 06

/Hans R. Troesch/
Hans R. Troesch
Reg. No. 36,950

Customer No. 32864
Fish & Richardson P.C.
Telephone: (650) 839-5070
Facsimile: (650) 839-5071